Recent Improvements on Cavity-Based Operators for RANS Mesh Adaptation

Adrien Loseille

Recent Improvements on Cavity-Based Operators for RANS Mesh Adaptation

Adrien Loseille[1)]
GAMMA3 Team, INRIA Saclay Ile-de-France, Palaiseau, France

If anisotropic mesh adaptation has been a reliable tool to predict inviscid flows, its use with viscous flows at high Reynolds number remains a tedious task. Indeed many issues tends to limit the efficiency of standard remeshing algorithms based on local modifications. First, the high Reynolds number require to handle a very high level of anisotropy $O(1:10^6)$ near the geometry. In the range of anisotropy, interpolation of metric fields or the projection on geometry are typical components that may fail during an adaptive step. The need for high-resolution near the geometry imposes to use an accurate geometry description, and optimally, be linked to a continuous CAD geometries. However, the boundary layer sizing may become smaller than typical CAD tolerance. We present a simple hierarchical geometry approximation where the newly created points are projected linearly, then using a cubic approximation then the CAD data. Finally, the accuracy, speed of convergence of the flow solver highly depends on the topology of the grids. Typical quasi-structured grids are preferred in the boundary layer while this kind of grids are complicated to generate with typical anisotropic meshing algorithm. We discuss in this paper, new developments in metric-orthogonal approach where an advancing points techniques is used to propose new points. Then these newly created points are inserted by using the cavity operator.

I. Introduction

Over the pas decade, an increasing interest has emerged to validate the numerical solutions of the Navier Stokes equations on complex geometries. The main idea was to verify that, at least, asymptotic convergence rate was obtained for grids of practical sizes. Many (successful) workshops have been organized to validate flow solver convergence, let's mention the Drag Prediction workshop and the High Lift prediction workshop, the Sonic Boom prediction. For each case, a very However, the $h/2$ refinement pattern is something that is barely tractable as the grids of the grids is multiplied by 8 at each step. Consequently, it seems necessary to investigate adaptive techniques wehre the refinement is controlled by a local error, taking advantage of the high directionality of the flows, to refined the grids at different speeds. Anisotropic mesh adaptation is one solution to address theoretically the asymptotic convergence issue.

II. Cavity operators

Cavity-based operators is a generic purpose adaptive mesh generator dealing with 2D, 3D and surface mesh generation. It belongs to the class of metric-based mesh generator [1–5] which aims at generating a unit mesh with respect to a prescribed metric field $\mathcal{M}$. A mesh is said to be unit when composed of almost unit-length edges and unit-volume element. The length of an edge AB in $\mathcal{M}$ is evaluated with:

$$\ell_{\mathcal{M}}(AB) \quad = \quad \int_0^1 \sqrt{t\,AB\,\mathcal{M}((1-t)A+tB)\,AB}\,\mathrm{d}t,$$

while the volume is given by $|K|_{\mathcal{M}} = \sqrt{\det \mathcal{M}}\,|K|$, where $|K|$ is the Euclidean volume of K. From a practical point of view, the volume and length requirements are combined into a quality function defined by :

$$Q_{\mathcal{M}}(K) = \frac{36}{3^{\frac{1}{3}}} \frac{\sum_{i=1}^{6} \ell^2_{\mathcal{M}}(\mathbf{e}_i)}{|K|^{\frac{2}{3}}_{\mathcal{M}}} \in [1,\infty],$$

[1)]Researcher, adrien.loseille@inria.fr

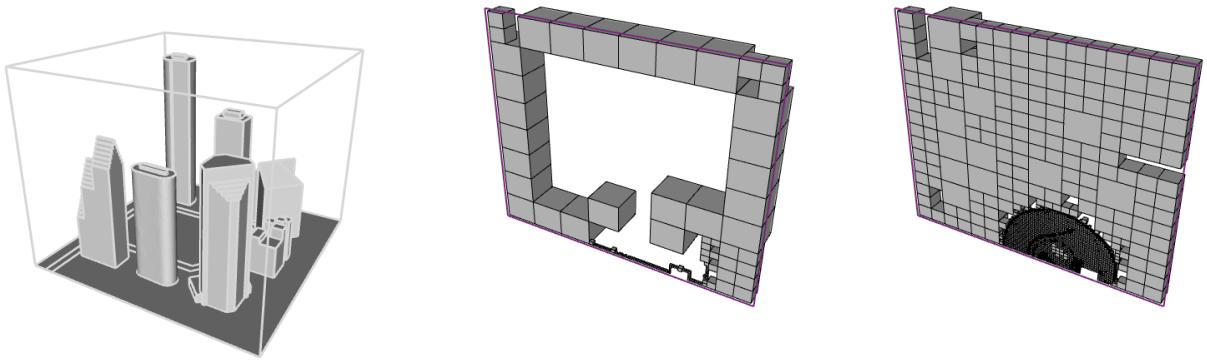


Fig. 1 Illustration of the filtering: from left to right, the geometry of generic city, the initial octree containing only the surface points, the final octree on the final set of points.

where $\{\mathbf{e}_i\}_{i=1,6}$ are the edges of element K. A perfect element has a quality of 1. The generation of a unit mesh is decomposed into two steps that are described below:

1) Generate a unit-mesh : The mesh modification operators are used in the goal to optimize the length of the edges in $\mathcal{M}$.
2) Optimization: The mesh modification operators are used to improve the quality $Q_{\mathcal{M}}$.

A. Generation of a unit mesh

The scope of this step is to obtain a mesh where the lengths of the edges are in $[\frac{1}{\sqrt{2}}, \sqrt{2}]$. This procedure is composed of 3 phases: collapse, creation of new points, anisotropic filtering and insertion.

Collapse. For this phase, an iterative procedure is used. The current mesh is iteratively scanned and while there exists an edge with a length lower than $1/\sqrt{2}$, try to collapse the edge. At the end of the process, all edges must have a length greater than $1/\sqrt{2}$. During all the following phases, the collapse is never used again.

Creation of edges. In this phase, we create the set of points that would be needed to decompose all long edges in segments having a length close to one in the metric. As for the collapse, the algorithm consists in scanning the current mesh and while there exists an edge with a length greater than 2, create one or multiple points. During this phase, the topology of the mesh is kept unchanged so that the points are not inserted. Indeed, neighboring edges can generate similar points or points very close to each other, so it is important to filter out the points that are too close (in the metric). For that, we define the anisotropic filtering.

Anisotropic filtering and insertion. In this phase, the length between the points created in the previous phase is checked and only a subset of points are inserted. For the filtering, we use an octree of points. Each octant can contain up to 10 points before being subdivided. Initially, the octree contains the surface points and the volume points remaining from the collapse phase. To validate the insertion of a point, we first check the distances between every points that are in the octant containing the point to be inserted. If no rejection occurs, then the current octant is intersected with the bounding box of the metric. All the intersected octants are checked starting from the octants closer to the point being inserted. Then, each point that is accepted for insertion is inserted in the octree along with its metric. At the end of the filtering, whatever the connectivity generated by the insertor the edges will have an admissible length (as the length was checked in every direction with the octree). This property prevents us from having to perform additional collapses that is the most costly operator. An octree is illustrated in figure 1.

B. Optimization of the mesh

During the phase, only the topology of the mesh is modified by using edges or faces swaps, see [6] for the details of these operators. The only constraint is to make sure that the quality in the metric is strictly improved at each application of a swap.

For all the previous phases, standard operators can be used [6]. However, in our approach, we use the cavity-based version for each of them. We show in this next section that this choice speeds up the CPU time of the remeshing by minimizing the number of rejections of each operator. We show also that one call of a cavity operator may be equivalent to a combination of several simple operator due to the use of cavity correction algorithm.

C. Cavity-based operators

A complete mesh generation or mesh adaptation process usually requires a large number of operators: Delaunay insertion, edge-face-element point insertion, edge collapse, point smoothing, face/edge swaps, etc. Independently of the complexity of the geometry, the more operators are involved in a remeshing process, the less robust the process may become. Consequently, the multiplication of operators implies additional difficulties in maintaining, improving and parallelizing a code. In [8], a unique cavity-based operator has been introduced which embeds all the aforementioned operators. This unique operator is used at each step of the process for surface and volume remeshing.

The cavity-based operator is inspired from incremental Delaunay methods [9–11] where the current mesh $\mathcal{H}_k$ is modified iteratively through sequences of point insertion. The insertion of a point P can be written:

$$\mathcal{H}_{k+1} = \mathcal{H}_k - C_P + \mathcal{B}_P, \tag{1}$$

where, for the Delaunay insertion, the cavity C_P is the set of elements of $\mathcal{H}_k$ such that P is contained in their circumsphere and $\mathcal{B}_P$ is the ball of P, i.e., the set of new elements having P as vertex. These elements are created by connecting P to the set of the boundary faces of C_P.

In [8], each meshing operator is equivalent to a node (re)insertion inside a cavity. For each operator, we just have to define judiciously which node P to (re)insert and which set of volume and surface elements will form the cavity C where point P will be reconnected with $\mathcal{R}_P$:

$$\mathcal{H}_{k+1} = \mathcal{H}_k - C + \mathcal{R}_P. \tag{2}$$

Note that if $\mathcal{H}_k$ is a valid mesh (only composed of elements of positive volume) then $\mathcal{H}_{k+1}$ will be valid if and only if C is connected (through internal faces of tetrahedron) and $\mathcal{R}_P$ generates only valid elements. In Figure 2, we list the initial cavity choice along with the point to (re)insert for the collapse, insertion and swap. As it, the cavity operators are equivalent to their standard counterparts. However, using the cavity formalism allows to easily modify the cavity to enforce automatically the operator. The cavity enlargement correction is one example of such correction and is given in Algorithm 1. The basic idea is to enlarge the cavity to make sure that C_P becomes valid. To illustrate this feature, we consider a simple 2D example where we want to relocate a point A to a new position A_{new}, see Figure 3. Given the initial configuration, we see that a collapse, then a swap and finally a point-smoothing is needed to actually move A to A_{new}. To do this, 4 volumes are computed for the collapse, 2 for the swap and finally 7 for the point-smoothing. Then, if we use the cavity version, the initial cavity has 2 negatives faces (in red in Figure 3, bottom). Using the cavity enlargement algorithm 1, a valid cavity is found in 3 enlargement iterations. To build the final C_P, 4 volumes are computed with the initial cavity, 4 for the first iterations, 2 for the second and 2 for the third. The cost of using the cavity moving is 12 volumes computations whereas 13 volumes are needed with the standard operators. The most interesting feature is that the cavity operator creates automatically the combination of simple operators without the need to know in practice the sequence. From a practical point of view, only one operator is used for the meshing operators.

The use of the previous cavity-based operators allows us to design a remeshing algorithm that has a linear complexity in time with respect to the required work (sum of the number of collapses and insertions). On a typical laptop computer Intel Core I7 at 2.7 GHz, the speed for the (cavity-based) collapse is around 20 000 points removed per second and the speed for the insertion is also around 20 000 points or equivalently 120 000 elements inserted per second. Both estimates hold in an anisotropic context [12].

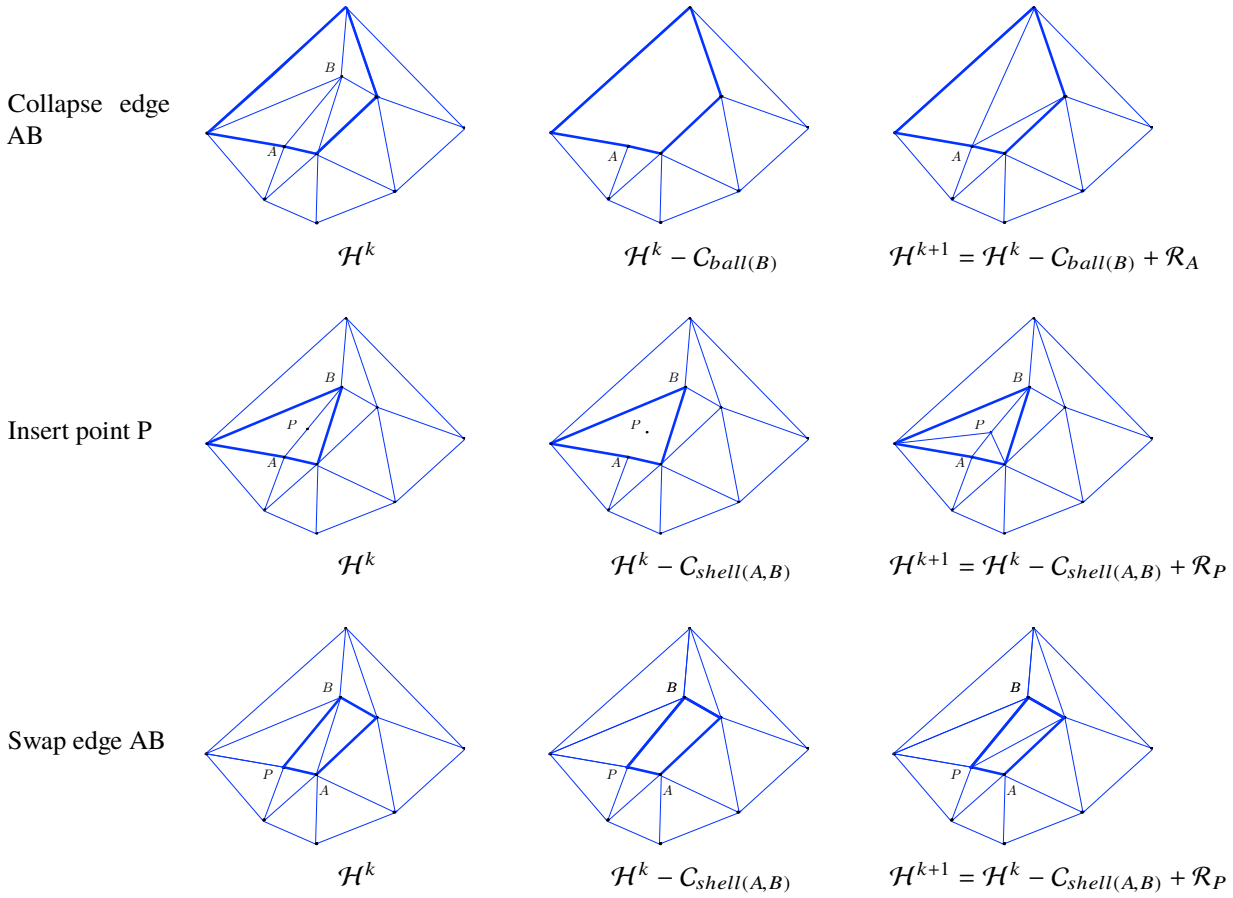


Fig. 2 Three 2D meshing operators reinterpreted as a cavity-based operator with an appropriate choice of the point to (re)insert and cavity to remesh. From top to bottom, the collapse, insertion and swap operators.

Algorithm 1 Cavity enlargement for (re)insertion of P

Volume Part:

For each K in C_P
  For each face $[A, B, C]$ such that $P \notin [A, B, C]$:
    if $volume(A, B, C, P) < 0$, then
      if P is a surface point then **reject**
      else add neighboring tetrahedron to C_P
      endif
    endif
  EndFor
EndFor

if C_P is modified goto **Volume Part**.

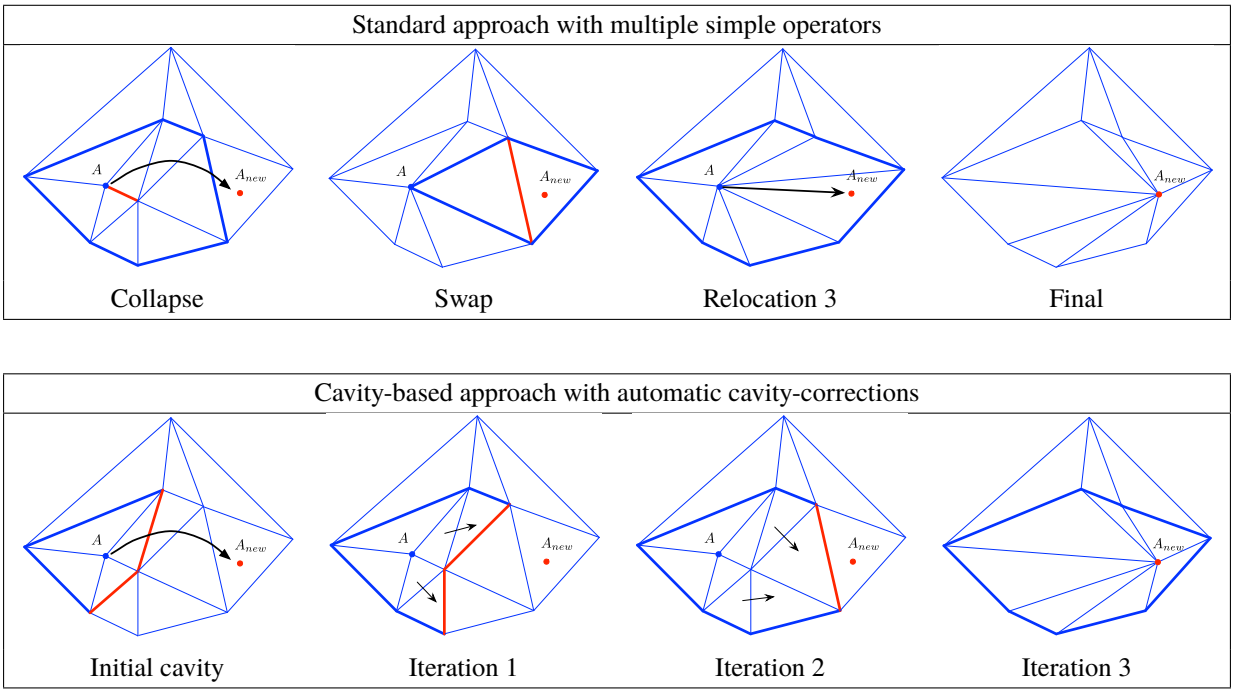


Fig. 3 Illustration of the relocation of point A to new position A_{new}. Top, if standard operators are used, the following sequence has to be applied: collapse, swap, relocation. Bottom, with the cavity enlargement, 3 enlargement iterations are needed to perform the operation.

III. CAD inverse projection

Inverse CAD projection usually arises when the link to CAD is missing or when high-order projection is required. In the context of mesh adaptation, working only in the parametric space is usually impossible for 3D mesh generation. It is then necessary to project back the point to the surface using the CAD information. Usually, CAD kernels provide such a functionality but it is not intended to be efficient for recurrent inverse projection. Here we present a simple procedure to perform inverse projection and CAD topology recovery.

The procedure first compute the inverse projection of inner points of patch. The topology recovery and the projection of line point is done in a second step. Fast inverse projection relies on a local Newton search. Starting from an initial (u_0, v_0) guess, we seek for the closest point $\sigma(u, v)$ to the surface so that:

$$\begin{cases} g(u,v) = (\sigma(u,v) - P) \cdot \sigma_u(u,v) = 0 \\ f(u,v) = (\sigma(u,v) - P) \cdot \sigma_v(u,v) = 0. \end{cases}$$

The critical point is to find a starting good candidate (u_0, v_0). To do so, we build an octree of points subdividing the parametric space. The Δu and Δv spacing is deduced from the degree of the NURBS. In a second step, local optimization is done. For each point, if the neigboring point has a lower distance from the initial point to the surface, the local optimization procedure is re-run with (u_0, v_0) being the (u, v) of the neighboring point.

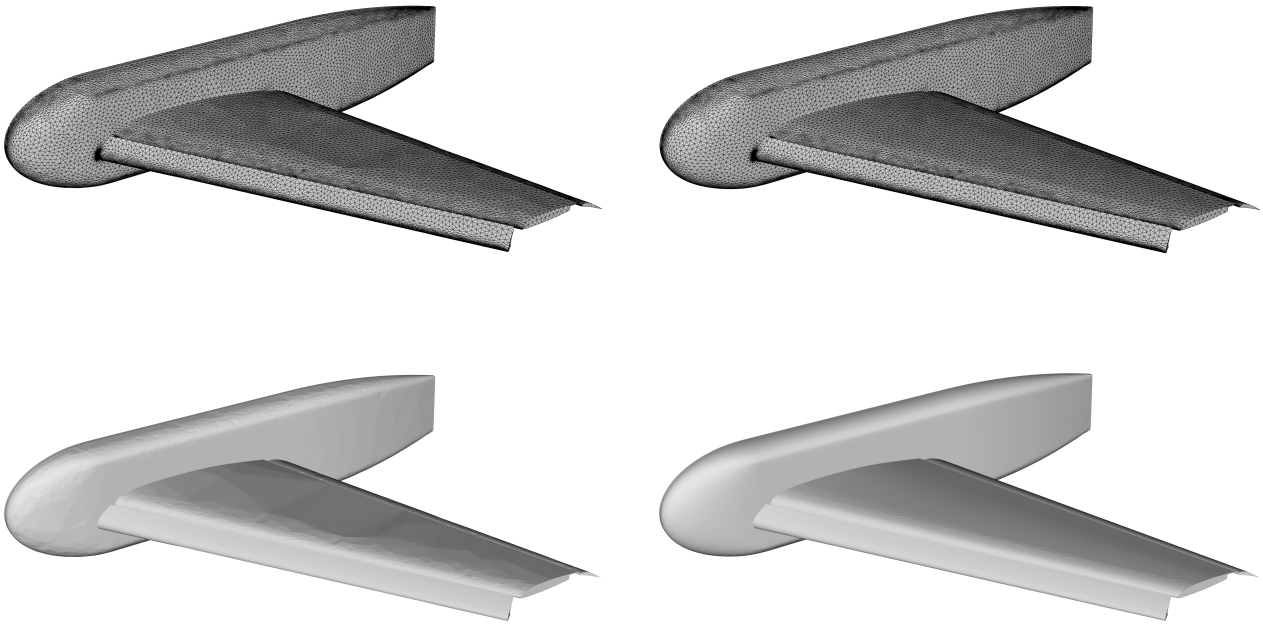


Fig. 4 Illustration of CAD projection on the first high-lift geometry.

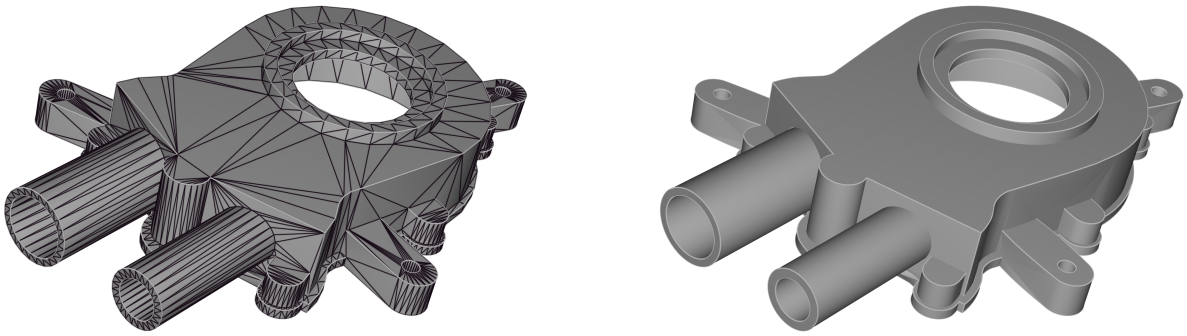
IV. Hierarchical geometry approximation

At first, a continuous description of a geometry, by using a CAD file in .igs or .stp file for instance, seems to be the most appropriate way to project points back the geometry. However, CAD files usually contained several impediments that may be

- wrong tolerances may appeared due to multiple of translation of the initial files from (often closed) format to another,
- NURBS of very high degrees leading to wiggled surface, coming the conversion of a discrete sample to a continuous description,
- tolerance that are bigger than the required sizes, leading to invalid CAD normals of points coordinates
- the topology of the CAD may be missing or wrong due to sizing imposed in the CAD (by splitting a continuous curves),
- ...

Consequently, it is mandatory to use a reliable geometry description. Using a fine linear tessellation is not necessary sufficient, as the size during an adaptive refinement may be of smaller size. The same observation holds for continuous CAD data, when the tolerances (when surface are intersected or trimmed) is bigger that the current required sizes. A simple workaround is then to consider, an intermediate level, in our case a $\mathbb{P}_3$ surface description that is recovered from a discrete mesh or from a discrete mesh with projected CAD normals on it.

The choice of approximating the surface with $\mathbb{P}_3$ triangle is inherited from the need to impose a $\mathcal{G}_1$ continuity of the vertex of the initial $\mathbb{P}_1$ mesh. The central node is then used to minimize the edge discontinuity, from applying subsequent smoothing steps.



P1 P3-recovered

Fig. 5 Example of P3 mesh recovered from a STL description of a geometry.

V. On boundary layers mesh generation

One of the limitation that limits the use of anisotropic mesh adaptation for RANS simulation regards the quality of the grids near the boundary layer profile. The typical meshing guidelines usually requires to generate a quasi-structured mesh extruded in the normal direction. This is required to fulfill the normal spacing. In a completely unstructured context, the high level of anisotropy implies also a very bad quality, in term of angles and alignment, in the area. In that respect, Finite Element flow solver (as SUPG based solver) seem to be less sensitive to the quality of the grids than mixed Finite Volume solvers.

To generate quasi-structured grids, a cavity based operator can be used. The main modification consists in keeping a list of element composing the boundary layer. This list of elements needs to be removed from the initial cavity:

$$\mathcal{H}_{k+1} = \mathcal{H}_k - (\mathcal{C}_P - \mathcal{K}) + \mathcal{B}_P, \tag{3}$$

where $(\mathcal{C}_P - \mathcal{K})$ has to be connected by faces. However in this approach, it is also complex to provide adaptivity functionality.

In order to favor orthogonality and metric alignment of the final mesh, a frontal approach is used. However, contrary to standard frontal approaches, we use a front of vertices instead of a front of faces. From a practical point of view, the new points are proposed by vertex and not by face. In an anisotropic context, the new points depend only on the eigenvectors and eigenvalues of the metric of the front point. The initial front of points is given by the list of the surface points. Given a point $\mathbf{x}_o$ and its metric $\mathcal{M}_o$ of the current front with eigenvectors $(\mathbf{u}_i)_{i=1,3}$ and eigenvalues $(\lambda_i)_{i=1,3}$, six points are proposed:

$$\mathbf{x}_i = \mathbf{x_o} \pm \lambda_i^{-\frac{1}{2}} \mathbf{u}_i. \tag{4}$$

When the metric is isotropic, we force the eigenvectors to be aligned with the natural axis of $\mathbb{R}^3$. Note that these points are just a first guess and several additional checks are performed before trying insertion. The first check consists in verifying that the new points are in the current volume mesh by using a simple mesh localization algorithm. This check is also performed on the background mesh. The back mesh localization also provides the metric $\mathcal{M}_i$ of $\mathbf{x}_i$.

In order to take into account the variation of the metric, the final position of $\mathbf{x}_i$ and metric $\mathbf{x}_i$ is updated. The procedure is based on a dichotomy along the segment $[\mathbf{x}_o, \mathbf{x}_i]$ in order to make sure that the Riemannian length evaluation of the vector $[\mathbf{x}_o\mathbf{x}_i]$ is unit:

$$\int_0^1 \sqrt{{}^t[\mathbf{x}_o\mathbf{x}_i]\,\mathcal{M}(t)\,[\mathbf{x}_o\mathbf{x}_i]}\,\mathrm{d}t = 1,$$

where $\mathcal{M}(t)$ is a geometric interpolation between metrics $\mathcal{M}(0) = \mathcal{M}_o$ and $\mathcal{M}(1) = \mathcal{M}_i$. Note that the original guess (4) only guarantees:

$${}^t[\mathbf{x}_o\mathbf{x}_i]\,\mathcal{M}_o\,[\mathbf{x}_o\mathbf{x}_i] = 1.$$

Consequently, we seek for an optimal point $\mathbf{x}_{opt}$ with back-mesh interpolated metric $\mathcal{M}_{opt}$ lying along the initial direction $\mathbf{x}_o\mathbf{x}_i$. Note that we need to iterate, because we interpolate the metric from the background mesh. If $\mathcal{M}_{opt}$ were interpolated by $\mathcal{M}_o$ and $\mathcal{M}_i$, an analytical formula exists depending on the metric interpolation scheme used. This list of points is then filtered in order to suppress from insertion points that are too close in the distance computed in the metric. The filtering process gives the list of points to be inserted. This list of points defines the next front. This algorithm is applied until the list of points to be inserted becomes empty.

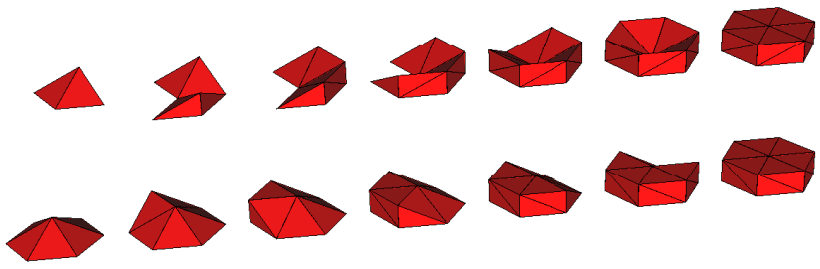


Fig. 6 *Tow examples of the automatic process of prisms creation around a point. The final decomposition of prisms (in tetrahedra) depends on the order of the insertion of points.*

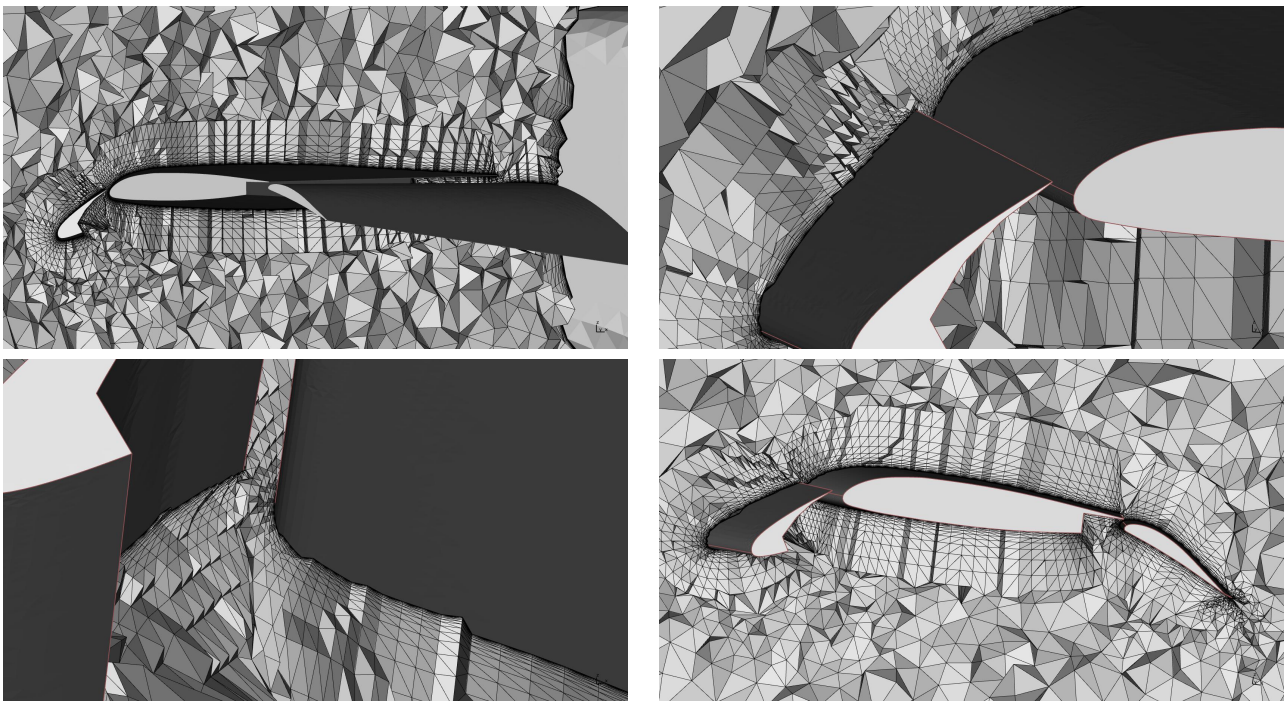


Fig. 7 Boundary layer mesh generation using the cavity-based operator on the high-lift geometry.

We illustrate in 2D the different set of points that can be naturally deduced from a metric field, see Figure 8. For boundary layer, we prefer to use the orthogonal approach where the points are created directly along the eigenvectors of the metric (that are naturally orthogonal). On a simple square domain with a circular metric, see Figure 9, we illustrate the advancing-point algorithm. In Figure 10 (top right), we show the list of points created when using Algorithm [**?**]. We can clearly see the alignment of the edges, Figure 10 (bottom right), in the final mesh when this points are inserted in comparison with the standard algorithm, Figure 10 (bottom left). To illustrate its used for boundary layer, we first illustrate its application in capturing the velocity profile of the flat plate test case of the the TMR website, see Figure 12 for the boundary conditions. In figure 12, we compare an adaptive computation (taking all CFD physical fields as sensor) when using standard adaptive technique and metric-orthogonal approach. In that case, as the metric computed from the error estimate has strong alignment feature, the resulting mesh looks like a boundary layer mesh. The second example is an inviscid flow around a Falcon geometry. In that case, we are interested in capturing the wakes and wing tip vortices. As for the boundary layer case, as the metric has strong alignment property, the resulting mesh follows the orientation of the input metric field, see Figure 13.

Algorithm 2 Metric-orthogonal algorithm

Advancing-point:

1. Pop the first heap list entry, creates $P_{new} = P_i \pm h_k \, \mathbf{u}_k$
2. Update length/position according of P_{new} to Riemannian metric field.
2. Metric-based length filtering using point octree, add P_{new} for insertion
3. Update the heap list with $(P_{new}, h_k, \mathbf{u}_k)_k$
4. If the heap list is not empty goto 2

Insertion:

1. Use cavity-based operators

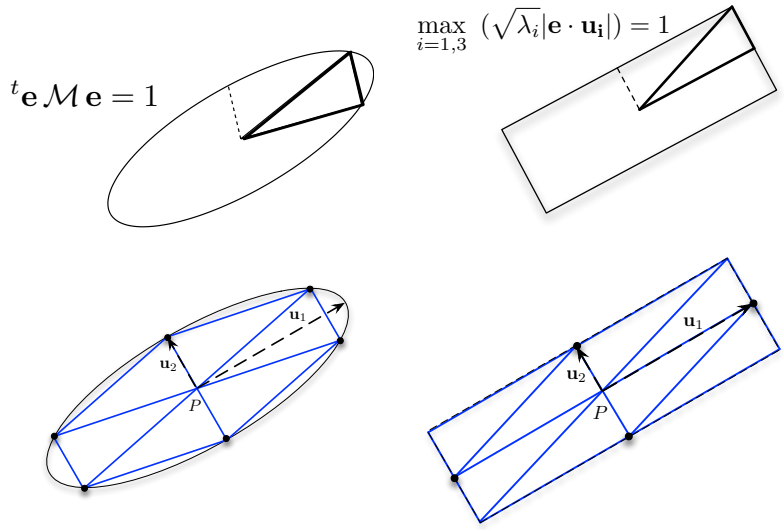


Fig. 8 Illustration in how optimal points are created with respect to the metric, with a metric-aligned (left) and metric-orthogonal (right).

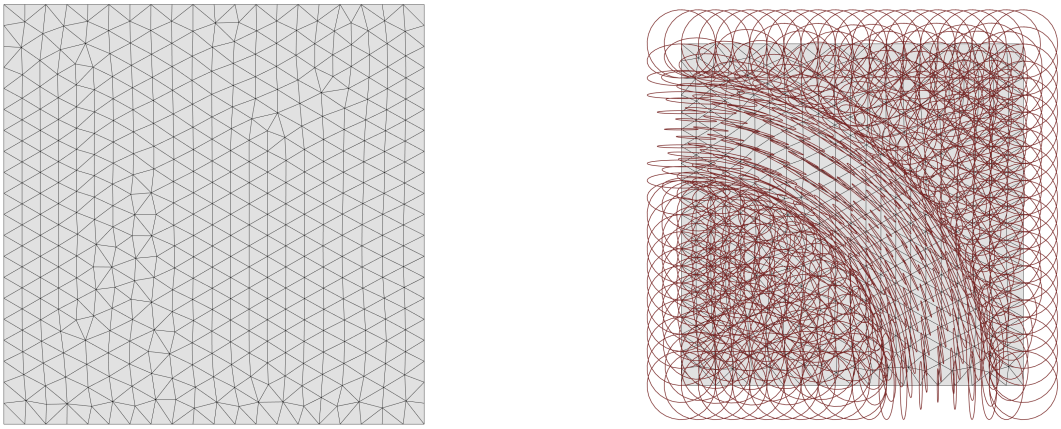
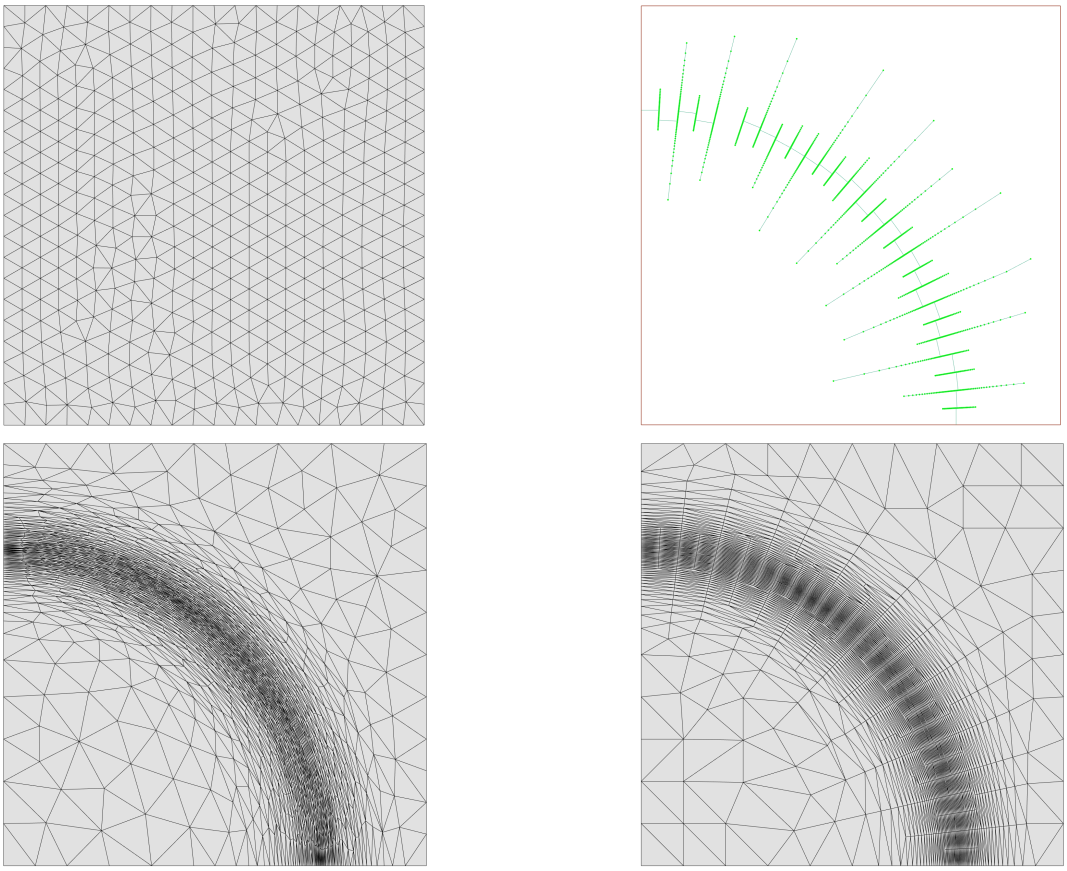


Fig. 9 Simple domain and initial mesh (left) equipped with a cylindrical metric field (right).



Standard Advancing-point

Fig. 10 Initial mesh (top left) and list of aligned-point to be inserted (top right), final mesh obtained with a standard procedure (bottom left) and final mesh obtained with the metric-orthogonal (bottom right).

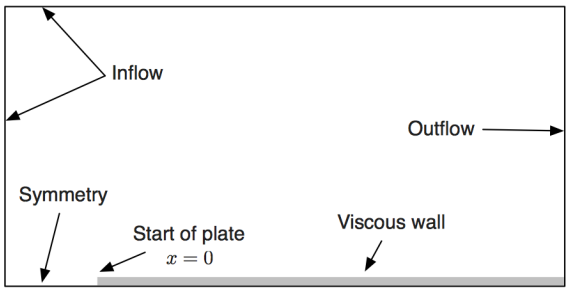


Fig. 11 Blasius flat plate test case description.

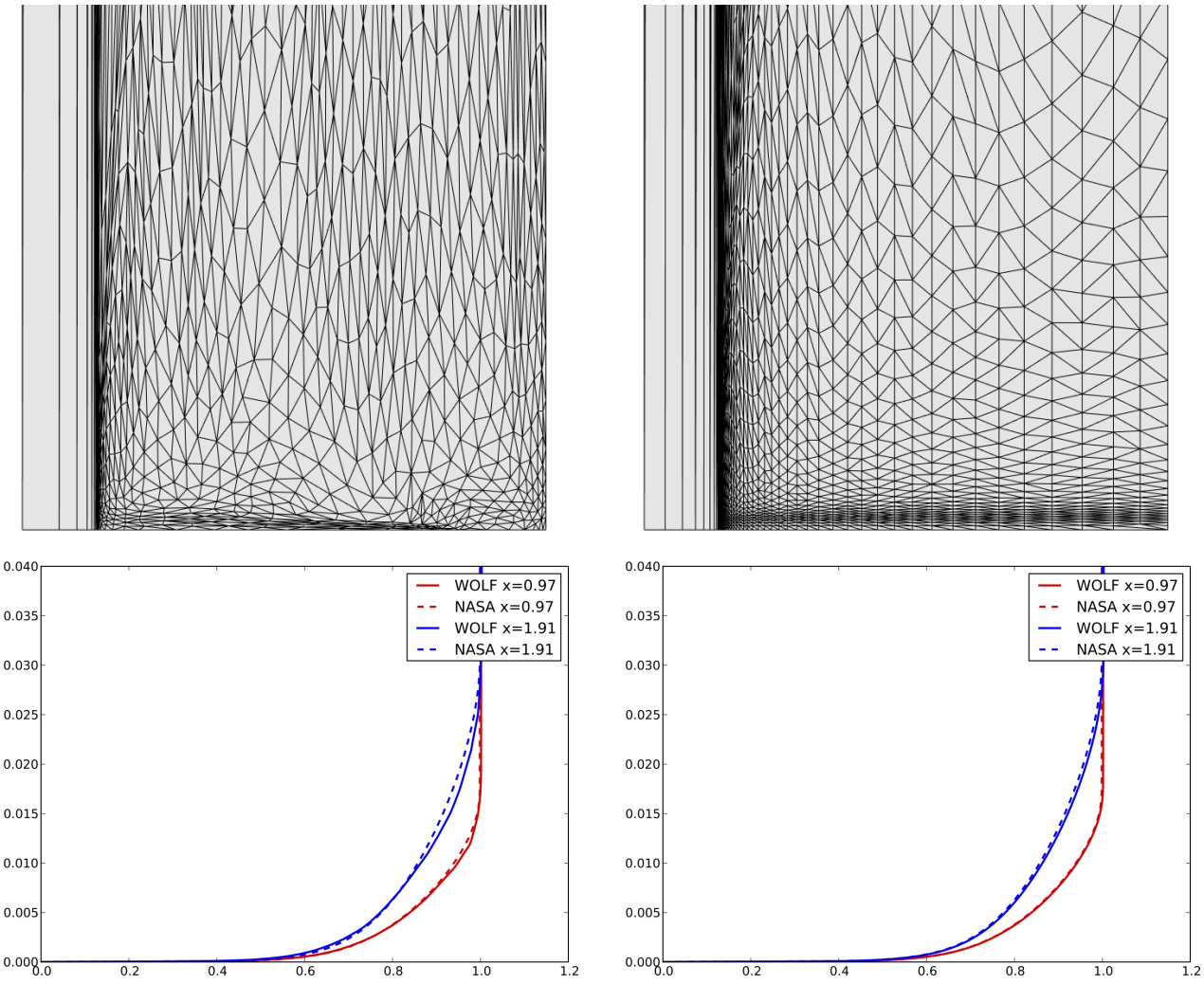


Fig. 12 Blasius flat plate test case with standard anisotropic procedure (left) and metric-orthogonal (right). Top, mesh near the stagnation point with a enlargement factor of 1000 in the y direction.

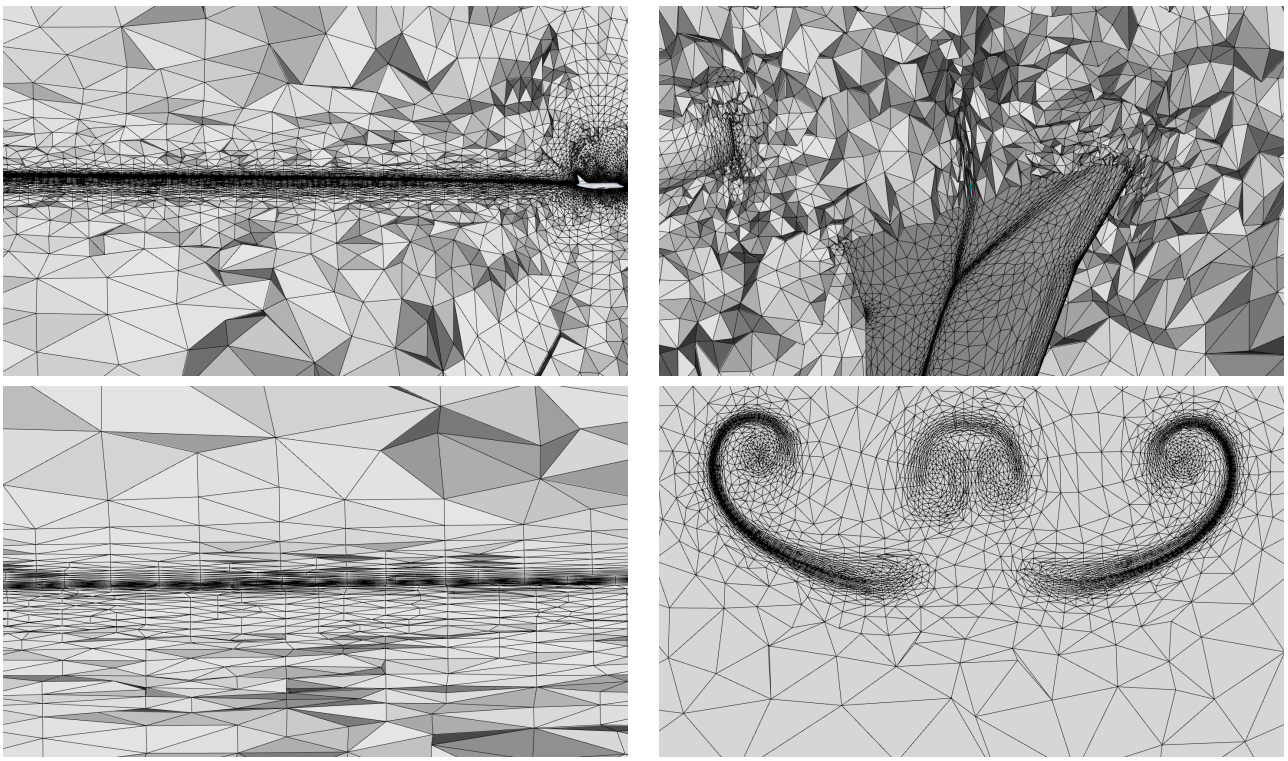


Fig. 13 Illustration of the a metric-orthogonal mesh refinement in capturing the wake of transsonic inviscid falcon.

Conclusion

In this paper, we have reviewed all the components involved in anisotropic meshing software. It is based on the unique cavity-based operator to avoid to and take advantage of cavity correction to enforce an operation. For the geometry description, an hierarchical approach is used where an initial point is created first on a linear mesh and then projected to cubic mesh and then to the CAD coordinates. The idea consists in trying the best approximation and then downgrading to a less accurate approximation (down to the linear one) in case of failure. Then to improve the quality of standard anisotropic meshing algorithm, an advancing-point is used coupled an multi-level cavity-based insertion. This allows to automatically favor the creation of quasi-structured grids in the boundary layer.

References

[1] Coupez, T., "Génération de maillages et adaptation de maillage par optimisation locale," *Revue Européenne des Éléments Finis*, Vol. 9, 2000, pp. 403–423.

[2] Dobrzynski, C., and Frey, P., "Anisotropic Delaunay Mesh Adaptation for Unsteady Simulations," *Proceedings of the 17th International Meshing Roundtable*, Springer, 2008, pp. 177–194.

[3] Li, X., Shephard, M., and Beal, M., "3D anisotropic mesh adaptation by mesh modification," *Computer Methods in Applied Mechanics and Engineering*, Vol. 194, No. 48-49, 2005, pp. 4915–4950.

[4] Loseille, A., and Löhner, R., "Adaptive anisotropic simulations in aerodynamics," *48th AIAA Aerospace Sciences Meeting*, AIAA Paper 2010-169, Orlando, FL, USA, 2010.

[5] Michal, T., and Krakos, J., "Anisotropic mesh adaptation through edge primitive operations," *50th AIAA Aerospace Sciences Meeting*, 2012.

[6] Frey, P., and George, P., *Mesh generation. Application to finite elements*, 2nd ed., ISTE Ltd and John Wiley & Sons, 2008.

[7] Loseille, A., and Löhner, R., "On 3D anisotropic local remeshing for surface, volume and boundary layers," *Proceedings of the 18th International Meshing Roundtable*, Springer, 2009, pp. 611–630.

[8] Loseille, A., and Menier, V., "Serial and Parallel Mesh Modification Through a Unique Cavity-Based Primitive," *Proceedings of the 22th International Meshing Roundtable*, Springer, 2013, pp. 541–558.

[9] Bowyer, A., "Computing Dirichlet tessellations," *Comput. J.*, Vol. 24, No. 2, 1981, pp. 162–166.

[10] Watson, D., "Computing the n-dimensional Delaunay tessellation with application to Voronoi polytopes," *Comput. J.*, Vol. 24, No. 2, 1981, pp. 167–172.

[11] Hermeline, F., "Triangulation automatique d'un polyèdre en dimension *N*," *RAIRO - Analyse numérique*, Vol. 16, No. 3, 1982, pp. 211–242.

[12] Loseille, A., "Metric-orthogonal anisotropic mesh generation," *Proceedings of the 23th International Meshing Roundtable, Procedia Engineering*, Vol. 82, 2014, pp. 403–415.